Description

SEMAPHORES WITH INTERRUPT MECHANISM

Cross-Reference to Related Application

This application claims the benefit of U.S. provisional application no. 60/266,002, filed February 2, 2001.

Technical Field

The invention generally relates to semaphores, and more particularly to semaphores in a multiprocessor system.

Background Art

Fig. 1 shows a multiprocessor system 100 of the prior art. The multiprocessor system 100 includes a system bus 110, a plurality of processors 120a, 120b coupled to the system bus 110, a main memory 130 coupled to the system bus 110, and a shared resource 150 coupled to the system bus 110. The main memory 130 includes a semaphore 140 which is used to monitor access to the shared resource 150 by the processors 120a, 120b. The processors 120a, 120b each include a register 125a, 125b.

In the prior art, the semaphore 140 is implemented by using a location in the main memory 130 whose content the processors 120a, 120b examine to determine whether the shared resource 150 is available for access. Assuming that initially the shared resource 150 is available for access, that is, both the processors 120a, 120b are not using the shared resource 150, then the content of the semaphore 140 would be a "0", indicating that the shared resource 150 is available for access. Assuming further that the processor 120a needs to access the shared resource 150, then the processor 120a would read the content of the semaphore 140 into its register 125a and then write a "1" into the semaphore 140. Design of the multiprocessor system 100 guarantees

that the reading of the semaphore 140 and writing a "1" into the semaphore 140 by the processor 120a, or by any other processor, constitutes one bus transaction. That is, the processor 120a seizes the system bus 110 continuously for both the reading and writing of the semaphore 140. Without this guarantee by design, a race condition may occur. A race condition occurs when at least two processors have access to a shared resource at the same time.

The processor 120a examines the copy of the content of the semaphore 140 which it receives in its register 125a and finds that the copy is a "0", indicating that the shared resource 150 is currently available for access. The processor 120a accesses the shared resource 150. Assume that while the processor 120a is using the shared resource 150, the processor 120b needs to access the shared resource 150. The processor 120b reads the content of the semaphore 140 into its register 125b and writes a "1" into the semaphore 140. Again, the design of the multiprocessor system 100 guarantees that the reading of the semaphore 140 and writing a "1" into the semaphore 140 by the processor 120b would constitute one bus transaction. The processor 120b then examines the copy of the content of the semaphore 140 which it receives in its register 125b and finds that the copy is a "1" indicating that the shared resource 150 is currently not available for access. The processor 120b would then execute a program loop that includes: (a) reading the content of the semaphore 140 into its register 125a, (b) writing a "1" into the semaphore 140, and (c) examining the copy of the content of the semaphore 140 which it receives in its register 125b. The processor 120b executes the program loop until the copy of the content of the semaphore 140 which the processor 120b receives in its register 125b is a "0".

When the processor 120a finishes using the shared resource 150, the processor 120a writes a "0" into

the semaphore 140. Recognizing that the content of the semaphore 140 becomes a "0", the processor 120b exits the program loop and accesses the shared resource 150. As a result, no race condition can occur.

While waiting for the shared resource 150 to be released by the processor 120a, the processor 120b keeps reading and writing the semaphore 140. This requires repeated use of the system bus 110. When the number of processors in the multiprocessor system 100 increases, a higher percentage of system bus cycles will be wasted on the processors reading and writing the semaphore 140 to determine when the shared resource 150 is released.

It is the object of the present invention to provide a method and apparatus in which semaphore implementation does not require repeated reading and writing the semaphore and, therefore, effectively increases the system bus throughput.

SUMMARY OF THE INVENTION

The above objects have been achieved by a digital system comprising a semaphore cell, an interrupt generation circuit coupled to the semaphore cell, and a processor coupled to the interrupt generation circuit. The semaphore cell is configured to have a first state and a second state, the first state of the semaphore cell indicating that a shared resource is available for access, and the second state of the semaphore cell indicating that the shared resource is unavailable for access. The interrupt generation circuit is configured to generate a semaphore interrupt signal to the processor if the semaphore cell changes from the second state to the first state and if the processor need to access the shared resource.

In another embodiment of the present invention, a method of using semaphores to monitor shared resource accesses is described. The method comprises providing a the semaphore cell configured to have a first state and a second state, the first state of the semaphore cell

indicating that the shared resource is available for access, and the second state of the semaphore cell indicating that the shared resource is unavailable for access. The method further comprises providing an interrupt generation circuit coupled to the semaphore cell and a processor coupled to the interrupt generation circuit. The method further comprises generating, with the interrupt generation circuit, a semaphore interrupt signal to the processor if the semaphore cell changes from the second state to the first state and if the processor need to access the shared resource.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a block diagram of a multiprocessor system 100 of prior art.

Fig. 2 shows a circuit diagram of a digital system 200 for implementing semaphores according to one embodiment of the present invention.

Fig. 3 shows a circuit diagram of one embodiment of the hardware semaphore cell 220a of the hardware semaphore register 220 and the corresponding semaphore interrupt cell 230a of the semaphore interrupt register 230 of Fig. 2.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

With reference to Fig. 2, the digital system 200 includes, illustratively, a system bus 210, processors 202a and 202b, shared resources 206a, 206b, and 206c, a hardware semaphore register 220, a semaphore interrupt register 230, semaphore interrupt enable registers 240 and 250, six AND gates 260a, 260b, 260c, 270a, 270b, and 270c, and two OR gates 280 and 290. The processors 202a and 202b include registers 204a and 204b, respectively.

The hardware semaphore register 220 includes, illustratively, three hardware semaphore cells 220a, 220b, and 220c. The hardware semaphore cells 220a, 220b, and 220c are used to monitor access to the shared

resources 206a, 206b, and 206c, respectively. In general, if a processor 202i (i = a or b) reads a hardware semaphore cell 220j (j = a, b, or c), the processor 202i receives the current content of the hardware semaphore cell 220j and the content of the hardware semaphore cell 220j becomes a "1". If a processor 202i writes a hardware semaphore cell 220j, the content of the hardware semaphore cell 220j always becomes a "0". In other words, if a processor 202i reads a hardware semaphore cell 220j which currently holds a "0", the processor 202i receives a "0" and the content of the hardware semaphore cell 220j becomes "1". If a processor 202i reads a hardware semaphore cell 220j which currently holds a "1", the processor 202i receives a "1" and the content of the hardware semaphore cell 220j remains "1".

The semaphore interrupt register 230 includes, illustratively, three semaphore interrupt cells 230a, 230b, and 230c, coupled to the hardware semaphore cells 220a, 220b, and 220c via connection line 223, 225, and 227, respectively. The semaphore interrupt enable register 240 includes, illustratively, three semaphore interrupt enable cells 240a, 240b, and 240c. The semaphore interrupt enable cells 240a, 240b, and 240c provides inputs to the AND gates 260a, 260b, and 260c via connection lines 243, 245, and 247, respectively. The AND gates 260a, 260b, and 260c also receive inputs from the semaphore interrupt cells 230a, 230b, and 230c via connection lines 233-283, 235-285, and 237-287. The AND gates 260a, 260b, and 260c generates three outputs to the OR gate 280 via connection lines 263, 265, and 267, respectively. The OR gate 280 generates a first semaphore interrupt signal to the processor 202a via connection line 281 if at least one of the AND gates 260a, 260b, and 260c generates a "1" to the OR gate 280.

Similarly, the semaphore interrupt enable register 250 includes, illustratively, three semaphore

interrupt enable cells 250a, 250b, and 250c. The semaphore interrupt enable cells 250a, 250b, and 250c provides inputs to the AND gates 270a, 270b, and 270c via connection lines 253, 255, and 257, respectively. The AND gates 270a, 270b, and 270c also receive inputs from the semaphore interrupt cells 230a, 230b, and 230c via connection lines 233-293, 235-295, and 237-297. The AND gates 270a, 270b, and 270c generates three outputs to the OR gate 290 via connection lines 273, 275, and 277, respectively. The OR gate 290 generates a second semaphore interrupt signal to the processor 202b via connection line 291 if at least one of the AND gates 270a, 270b, and 270c generates a "1" to the OR gate 290.

For illustration of the operation of the digital system 200, assume that, initially, the hardware semaphore cell 220a, the semaphore interrupt cell 230a, and the semaphore interrupt enable cells 240a and 250a all hold a "0". Assume further that the processor 202a needs to access the shared resource 206a. The processor 202a reads the hardware semaphore cell 220a into its register 204a. Reading the hardware semaphore cell 220a by the processor 202a automatically changes the content of the hardware semaphore cell 220a from "0" to "1". The processor 202a then examines the copy of the hardware semaphore cell 220a which it gets in its register 204a and finds that the copy is a "0" indicating that the shared resource 206a is currently available for access. The processor 202a then accesses the shared resource 206a.

Assume that while the processor 202a is using the shared resource 206a, the processor 202b needs to access the shared resource 206a. The processor 202b reads the hardware semaphore cell 220a into its register 204b. Any reading of the hardware semaphore cell 220a by any processor automatically sets the content of the hardware semaphore cell 220a to "1". Because the hardware semaphore cell 220a currently holds a "1", reading the hardware semaphore cell 220a by the processor

202b does not change this content of hardware semaphore cell 220a (still a "1"). The processor 202b then examines the copy of the hardware semaphore cell 220a which it gets in its register 204b and finds that the copy is a "1" indicating that the shared resource 206a is currently unavailable for access. The processor 202b sets the content of the semaphore interrupt enable cell 250a to "1" and then switches to another task. As a result, the AND gate 270a receives as an input a "1" from the semaphore interrupt enable cell 250a via connection line 253.

When the processor 202a no longer needs access to the shared resource 206a, the processor 202a writes a "0" into the hardware semaphore cell 220a. This causes the content of the semaphore interrupt cell 230a to change from "0" to "1". This content of "1" of the semaphore interrupt cell 230a propagates to the AND gate 270a as an input via connection line 233-293. In response, the AND gate 270a generates a "1" to the OR gate 290 which in turn generates a "1" as the second semaphore interrupt signal to the processor 202b causing an interrupt in the processor 202b.

This content of "1" of the semaphore interrupt cell 230a also propagates to the AND gate 260a as an input via connection line 233-283. However, because the semaphore interrupt enable cell 240a holds a "0", the other input of the AND gate 260a is a "0". As a result, the AND gate 260a generates a "0" to the OR gate 280 which in turn generates a "0". Therefore, no interrupt occurs in the processor 202a.

In response to the interrupt, the processor 202b services the interrupt by reading the contents of the semaphore interrupt register 230 and semaphore interrupt enable registers 250 via connection buses 239 and 259, respectively. Because both the semaphore interrupt cell 230a and the semaphore interrupt enable cell 250a hold a "1", processor 202b can determine that

the release of the corresponding shared resource 260a caused the interrupt. The processor 202b then writes a "0" to both the semaphore interrupt cell 230a and semaphore interrupt enable cell 250a via connection buses 239 and 259, respectively. The processor 202b then reads the hardware semaphore cell 220a into its register 204b. The reading of the hardware semaphore cell 220a by the processor 202b also changes to content of the hardware semaphore cell 220a from "0" to "1". The processor 202b then examines the copy of the hardware semaphore cell 220a which it gets in its register 204b and finds that the copy is a "0" indicating that the shared resource 206a is currently available for access. The processor 202b then accesses the shared resource 206a. In summary, the processor 202b does not have to repeatedly check the hardware semaphore cell 220a via the system bus 210 to determine if the shared resource 206a is released. As a result, the throughput of the system bus 210 is increased.

In a similar manner, the hardware semaphore cell 220b, the semaphore interrupt cell 230b, and the semaphore interrupt enable cells 240b and 250b are used to monitor access to the shared resource 206b by the processors 202a and 202b, respectively. Also in a similar manner, the hardware semaphore cell 220c, the semaphore interrupt cell 230c, and the semaphore interrupt enable cells 240c and 250c are used to monitor access to the shared resource 206c by the processors 202a and 202b, respectively.

With reference to Fig. 3, the hardware semaphore cell 220a and the semaphore interrupt cell 230a of Fig. 2 are shown in further detail according to one preferred embodiment. The hardware semaphore cell 220a includes an address decoder 310, a multiplexer 320, a D flip-flop 330, an AND gate 340, and a tri-state buffer 390. The Q output of the D flip-flop 330 holds the current content of the hardware semaphore cell 220a.

Assume the processor 202a (Fig. 2) reads the hardware semaphore cell 220a. The processor 202a reads the hardware semaphore cell 220a by putting a unique address of the hardware semaphore cell 220a on a semaphore address bus 313 and putting a "1" on a control line 315. In response, the address decoder 310 generates a "1" to the multiplexer 320 via a connection line 325 causing the multiplexer 320 to electrically connect the control line 315 to the D input of the D flip-flop 330 via connection line 329. As a result, the output Q of the D flip-flop 330 will hold a "1" in the next clock cycle. The AND gate 340 receives a "1" from the address decoder 310 via connection line 323. The AND gate 340 also receives a "1" from the control line 315. As a result, the AND gate 340 generates a "1" to the buffer 390 causing the buffer 390 to pass the current content of the D flip-flop 330 at the output Q of the D flip-flop 330 to the processor 202a via connection lines 393, 317, and the system bus 210. In summary, the reading of the hardware semaphore cell 220a by the processor 202a gives the processor 202a the current content of the hardware semaphore cell 220a and puts a "1" into the hardware semaphore cell 220a.

The semaphore interrupt cell 230a includes, illustratively, D flip-flops 350 and 380, an OR gate 360, and an AND gate 370. The Q output of the D flip-flop 350 holds the current content of the semaphore interrupt cell 230a. Assume that the hardware semaphore cell 220a currently holds a "1" indicating the processor 202a is using the shared resource 206a (Fig. 2) and that the semaphore interrupt cell 230a currently holds a "0". In other words, the Q output of the D flip-flop 330 currently holds a "1" and the Q output of the D flip-flop 350 currently holds a "0". The D input and the Q output of the D flip-flop 380 also hold a "1".

When the processor 202a finishes using the shared resource 206a, the processor 202a writes a "0"

into the hardware semaphore cell 220a. The processor 202a writes a "0" into the hardware semaphore cell 220a by putting the unique address of the hardware semaphore cell 220a on the semaphore address bus 313 and putting a "0" on the control line 315. The address decoder 310 generates a "1" to the multiplexer 320 causing the multiplexer 320 to pass the value "0" on the control line 315 to the D input of the D flip-flop 330. As a result, the hardware semaphore cell 220a will hold a "0" in the next clock cycle.

When the Q output of the D flip-flop 330 changes from "1" to "0", the output of the AND gate 370 changes from "0" to "1". As a result, the output of the OR gate 360 changes from "0" to "1", which is applied to the D input of the D flip-flop 350. In the next clock cycle, the semaphore interrupt cell 230a will hold a "1", which is applied to the AND gate 270a (Fig. 2) via connection line 233-293. In response, the AND gate 270a generates a "1" to the OR gate 290, assuming that the semaphore interrupt enable cell 250a has been set to "1" by the processor 202b. As a result, the OR gate 290 generates a "1" as the second semaphore interrupt signal to the processor 202b causing an interrupt in the processor 202b. In response to the interrupt, the processor 202b services the interrupt by determining which semaphore caused the interrupt. The processor 202b makes this determination by comparing the contents of the semaphore interrupt register 230 and semaphore interrupt enable register 250. Because both the semaphore interrupt cell 230a and the semaphore interrupt enable cell 250a hold a "1", the processor 202b can determine that the semaphore interrupt cell 230a caused the interrupt. In other words, the processor 202b can determine that the release of the shared resource 206a caused the interrupt. The processor 202b then writes a "0" to both the semaphore interrupt cell 230a and semaphore interrupt enable cell 250a via connection buses

239 and 259, respectively. The processor 202b then reads the hardware semaphore cell 220a into its register 204b. The reading of the hardware semaphore cell 220a by the processor 202b also changes to content of the hardware semaphore cell 220a from "0" to "1". The processor 202b then examines the copy of the hardware semaphore cell 220a which it gets in its register 204b and finds that the copy is a "0" indicating that the shared resource 206a is currently available for access. The processor 202b then accesses the shared resource 206a. In summary, in the present invention, the use of hardware semaphores coupled with interrupt mechanism avoids race conditions and repetitive use of the system bus for checking the contents of the hardware semaphores.

In one embodiment of the present invention, "software semaphores" may be used in place of hardware semaphores 220. The software semaphores may be registers like the hardware semaphores 220 but do not automatically change to a certain state (e.g., become "1") when being read by a processor. If software semaphores are used, the instruction set must include a special test-and-set instruction which reads the software semaphores and sets the software semaphores to "1" in one atomic action.

In another embodiment of the present invention, the digital system 200 may have M processors 202 and N shared resources 206. Accordingly, the digital system 200 has a hardware semaphore register 220 and a semaphore interrupt register 230 each of which has N cells corresponding to the N shared resources 206. The digital system 200 further has M semaphore interrupt enable registers 240, 250 corresponding to the M processors 202. Each semaphore interrupt enable register 240, 250 has N cells corresponding to the N shared resources 206. In the embodiments of the present invention described above, M = 2 and N = 3.